UNIVERSITÀ DEGLI STUDI
DI MILANO

Convolutional Neural Networks for binary image classification

Farnaz Sharbafi[1]

Abstract

In this project, a Convolutional Neural Network (CNN) was developed for binary image classification of cats and dogs dataset. The aim of the project is to build a model that accurately detects images of cats and dogs. Different network architectures and training parameters are experimented with, and their influence on the final predictive performance is presented.

The dataset used for training and testing consisted of images of cats and dogs. A 5-fold cross-validation is used to compute the risk estimates, and the reported cross-validated estimates are computed according to the zero-one loss. The results showed that the developed CCN model achieved high accuracy in classifying images of cats and dogs, demonstrating the effectiveness of CNNs in image classification tasks. Overall, this project highlights the importance of exploring different network architectures and training parameters to optimize the performance of CNNs in image classification tasks.

Keywords: Convolutional Neural Network (CNN), Binary image classification

Introduction

Convolutional Neural Networks (CNNs) are a type of artificial neural network commonly used in image processing and computer vision applications. They are designed to recognize patterns and features in images by mimicking the way the human brain processes visual information. They have shown impressive results in various applications, including object recognition, face detection, and medical image analysis (Krizhevsky et al., 2012; Zhang et al., 2016). CNNs are particularly well-suited for image classification

[1] Matriculation number: 964159

tasks due to their ability to automatically learn discriminative features from raw image data.

The architecture of a CNN typically consists of a series of convolutional layers, followed by pooling layers and fully connected layers. Convolutional layers apply filters to the input image to extract important features, while pooling layers reduce the spatial size of the feature maps. Fully connected layers then perform classification based on the features extracted by the previous layers. CNNs have also been extended to handle more complex tasks such as semantic segmentation, object detection, and image generation (Long et al., 2015; Ren et al., 2015).

CNNs have been shown to be highly effective in a number of image classification tasks, and have been outperformed human experts in certain cases. For example, in 2015, a CNN called RestNet-50 achieved top performance on the ImageNet Large Scale Visual Recognition Challenge, a benchmark dataset for image classification. ResNet-50 was able to classify images with an error rate of just 3.57%, outperforming human experts who had an error rate of 5.1% on the same dataset.

There have been many advancements in the field of CNNs in recent years, including the development of novel architectures such as InceptionNet, DenseNet, and EfficientNet. These architectures incorporate various techniques such as residual connections, dense connections, and efficient scaling to improve the performance of CNNs.

Overall, CNNs have proven to be a powerful tool in image classification and computer vision, and continue to be an active area of research.

In recent years, CNNs have been increasingly applied to binary image classification tasks such as cat vs dog classification (Jajodia, T., & Garg, P. (2019); Lee, Y. (2021)). Binary classification tasks involve categorizing an image into one of two classes, making them particularly useful in applications such as medical diagnosis and security surveillance.

In this work, a CNN for binary image classification of cats and dogs is developed and the model is experimented with different network architectures and training parameters to optimize its performance. We also employ 5-fold cross-validation to compute risk estimates using the zero-one loss function.

Dataset

The given dataset initially contains 25,000 images, where 12,500 images are of dogs and 12,500 images are of cats. These images are loaded into the Python environment using the OpenCV library, which is a computer vision library. During the loading process, the images are also converted into grayscale, which is a common technique used in image processing to reduce the complexity of the image. In addition, all of the images scaled down to 50x50 pixels in order to obtain the dataset with the same dimensions. Also, pixels are devided to 255 to have rescaled [0,1] values. However, there might be cases where an image cannot be loaded or resized, for example, if the file is corrupted or damaged. In such cases, the image is dropped from the list and not included in the final dataset. After this process, it is found that 24946 images have been successfully loaded and processed, and they are ready to be used for further analysis or machine learning tasks. It is important to note that the number of images dropped from the original dataset is relatively small compared to the total number of images, which means that the dataset is still representative and suitable for further analysis.

In the figure 1, 8 processed sample images are shown.

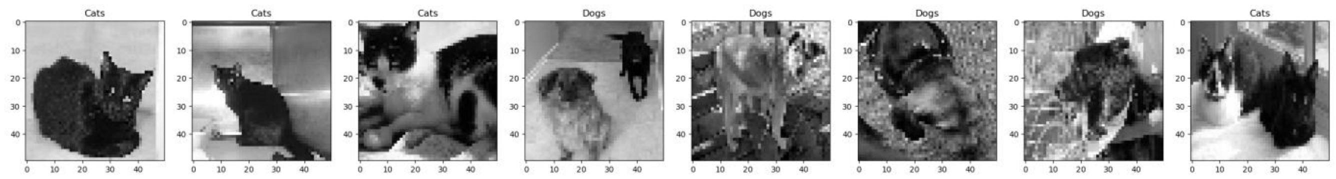


Figure 1: Cat and dog images after preprocessing steps

Deep learning and Neural Network

Deep learning is a subset of machine learning, which is based on artificial neural networks (ANNs) to model and solve complex problems. Neural networks are a collection of interconnected nodes (also called artificial neurons) that can receive input, process it, and produce output. The nodes are arranged in layers, and the connections between them have weights that are adjusted during the learning process.

Deep learning algorithms can automatically learn hierarchical representations of data, by composing multiple layers of nonlinear transformations. These representations can be used to solve a variety of problems, such as image classification, speech recognition, natural language processing, and many others.

In practice, a deep learning model is trained on a large dataset, by iteratively adjusting its weights to minimize a loss function that measures the difference between the predicted output and the true output. This is typically done using an optimization algorithm such as stochastic gradient descent (SGD). Once the model is trained, it can be used to make predictions on new data.

Deep learning has achieved state-of-the-art performance on many tasks, and has revolutionized fields such as computer vision, natural language processing, and robotics. However, it requires large amounts of data and computing power, and can be difficult to train and tune.

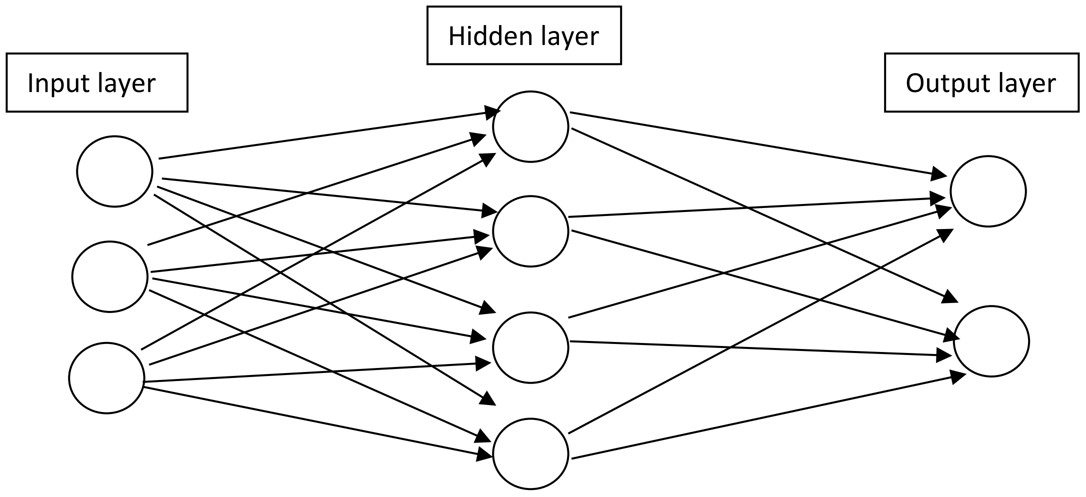


Figure 2: Structure of a netwrok

Neural networks can be explained mathematically as a composition of linear and non-linear transformations. In a simple neural network with a single hidden layer, let the input be a vector of size *n* and the output be a vector of size *m*. Let *X* be the input vector and *Y* be the output vector. The neural network takes the input *X* and produces the output *Y* through a sequence of computations.

The first computation is a linear transformation, which takes the input vector *X* and transforms it into a vector *Z* by multiplying it with a matrix *W* of size *n* times *p* and adding a bias vector *b* of size *p*:

$$Z = XW + b$$

where *p* is the number of neurons in the hidden layer.

The next computation applies a non-linear activation function sigma to the vector *Z* element-wise to produce a new vector *H*:

$$H = \sigma(Z)$$

The non-linear activation function introduces non-linearity into the neural network, which allows it to model complex relationships between the input and output.

The final computation is another linear transformation, which takes the vector *H* and transforms it into the output vector *Y* by multiplying it with a matrix *V* of size *p* times *m* and adding a bias vector *c* of size *m*:

$$Y = HV + c$$

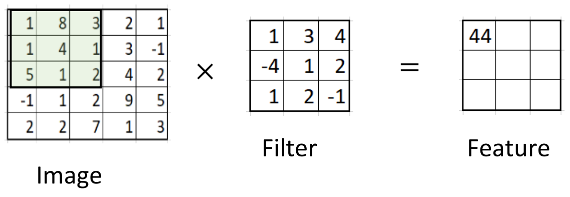
The matrices *W* and *V* are called weights, and the bias vectors *b* and *c* are called biases. These are the parameters of the neural network, which are learned from data during the training process.

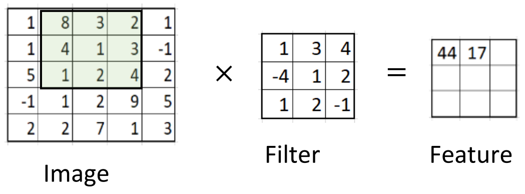
The entire neural network can be represented as a function *f(X; W, V, b, c)* that takes the input *X* and parameters *W*, *V*, *b*, *c* and produces the output *Y*. The objective of training a neural network is to find the optimal values of these parameters that minimize a given loss function between the predicted output and the true output. This is typically done using techniques such as backpropagation and stochastic gradient descent.

Layers

Convolutional layers are a fundamental component of convolutional neural networks (CNNs) and are used for feature extraction in image processing tasks. Mathematically, a convolutional layer applies a set of filters to the input image, producing a set of feature maps.

Let's consider an input image of size H x W x D, where H is the height, W is the width, and D is the number of channels (for example, D=1 for a gray scale color image). The convolutional layer consists of a set of filters, each of size F x F x D, where F is the filter size. Each filter is applied to the input image by sliding it over the image, computing the dot product of the filter values with the corresponding image pixels at each position. This operation is called a convolution, hence the name "convolutional layer."





$$\underset{\text{Image}}{\begin{array}{|c|c|c|c|c|}\hline 1 & 8 & 3 & 2 & 1 \\ \hline 1 & 4 & 1 & 3 & -1 \\ \hline 5 & 1 & 2 & 4 & 2 \\ \hline -1 & 1 & 2 & 9 & 5 \\ \hline 2 & 2 & 7 & 1 & 3 \\ \hline \end{array}} \times \underset{\text{Filter}}{\begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline -4 & 1 & 2 \\ \hline 1 & 2 & -1 \\ \hline \end{array}} = \underset{\text{Feature}}{\begin{array}{|c|c|c|}\hline 44 & 17 & 18 \\ \hline 1 & 21 & 21 \\ \hline 24 & 54 & 39 \\ \hline \end{array}}$$

Figure 3: Convolutional layer

The output of the convolution operation is a two-dimensional feature map, where each pixel in the map represents the dot product of the filter with a local patch of the input image. The size of the feature map depends on the size of the input image, the size of the filter, and the stride of the convolution (the amount by which the filter shifts at each step).

In practice, multiple filters are applied to the input image to produce multiple feature maps, each representing a different aspect of the image. These feature maps are then passed through an activation function, such as ReLU, to introduce nonlinearity.

The resulting feature maps are then typically downsampled using pooling layers, which reduce the spatial resolution of the feature maps while retaining the most important features. The main purpose of pooling layers is to reduce the spatial size of the input volume (i.e., the output of the previous convolutional layer), while retaining the most important features. This helps to reduce the computational complexity of the network and prevent overfitting.

44	21
54	54

Figure 4: MaxPooling layer operation for the previous example

After passing through the convolutional layers and pooling layers, the Flatten layer converts the output of the previous layer into a one-dimensional array. The dense layer connects each neuron in the layer to every neuron in the previous layer and performs a non-linear transformation of the input using weights, biases, and an activation function learned through training. After that classification will be performed on the input image.

Models

Model 1: This model contains two convolutional layers with 64 nodes followed by a maxpooling layer, flatten layer and two dense layers with 128 and 1 nodes.

Model 2: This model contains four convolutional layers (two blocks each of them two layers) with 32 and 64 nodes for each block, followed by a maxpooling layer (for each block), flatten layer and two dense layers with 128 and 1 nodes.

For each model the drop out layer is also added to see the performance of the model.

Model performance

In this project two model architectures are implemented. In the project, it can be specified that due to hardware limitations, the images were resized to a relatively small resolution (50X50) to accommodate the computational requirements. This resizing was necessary to ensure that the code could run smoothly on the available hardware resources. The resized images were used for training, validation, and testing purposes throughout the project. It should be noted that while the smaller image size may have affected the overall accuracy or performance of the image classification model, the primary focus was to develop a functional solution within the given constraints.

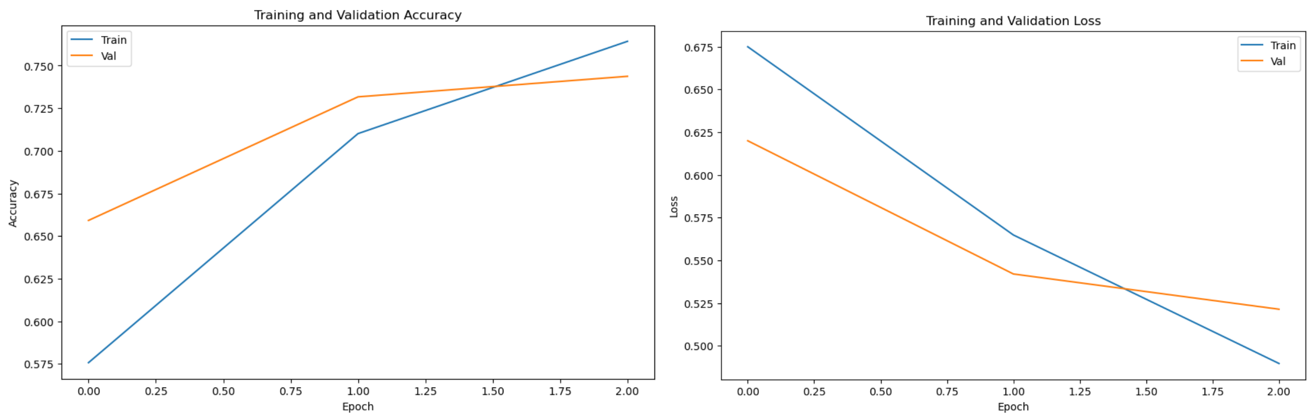


Figure 5. Model 1 accuracy and loss plot

Model 1	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss
Fold 1	0.6946	0.5723	0.7135	0.5509
Fold 2	0.6833	0.5817	0.7271	0.5426
Fold 3	0.6769	0.5849	0.6991	0.5751
Fold 4	0.6798	0.5831	0.6923	0.5730
Fold 5	0.7005	0.5593	0.7281	0.5460
Avg	**0.6870**	**0.5763**	**0.7120**	**0.5575**

Table 1. Model 1 Training and validation results in each fold

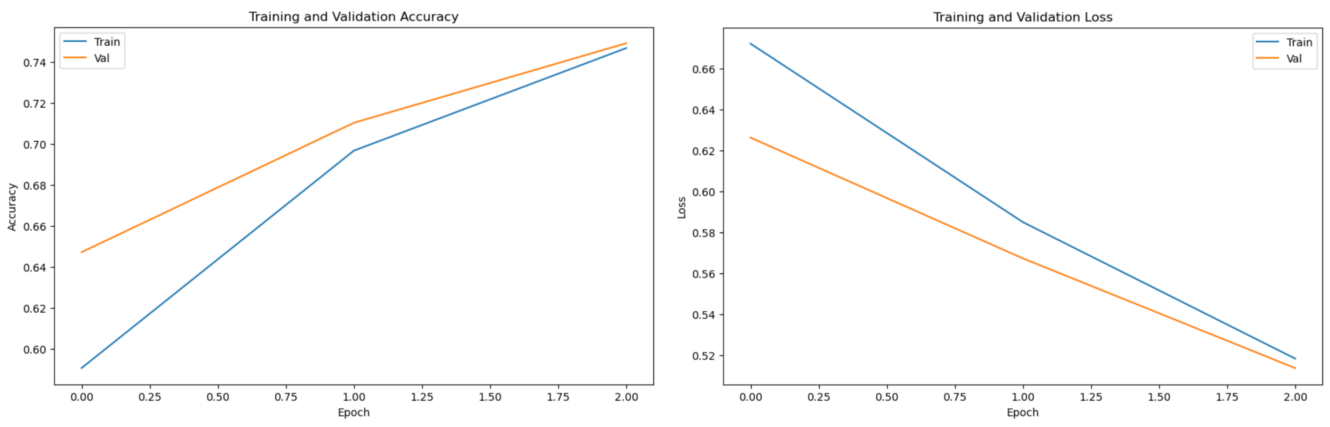


Figure 6. Model 1 with Dropout layer accuracy and loss plot

Model 1	**Train Accuracy**	**Train Loss**	**Validation Accuracy**	**Validation Loss**
Fold 1	0.6829	0.5857	0.7118	0.5598
Fold 2	0.6688	0.6005	0.6946	0.5768
Fold 3	0.6736	0.5953	0.6981	0.5758
Fold 4	0.6895	0.5774	0.7148	0.5532
Fold 5	0.6749	0.5996	0.6913	0.5791
Avg	**0.6779**	**0.5917**	**0.7021**	**0.5689**

Table 2. Model 1 with Dropout layer Training and validation results in each fold

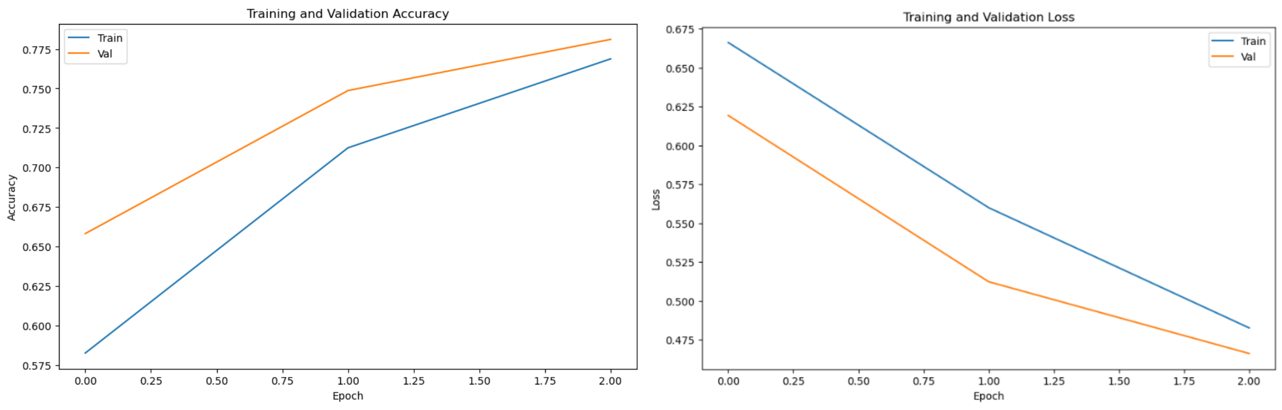


Figure 7. Model 2 accuracy and loss plot

Model 2	**Train Accuracy**	**Train Loss**	**Validation Accuracy**	**Validation Loss**
Fold 1	0.6707	0.5922	0.7046	0.5638
Fold 2	0.6805	0.5784	0.7352	0.5290

Fold 3	0.7080	0.5486	0.7386	0.5176
Fold 4	0.7004	0.5521	0.7459	0.5144
Fold 5	0.6804	0.5767	0.7225	0.5383
Avg	**0.6880**	**0.5696**	**0.7293**	**0.5326**

Table 3. Model 2 Training and validation results in each fold

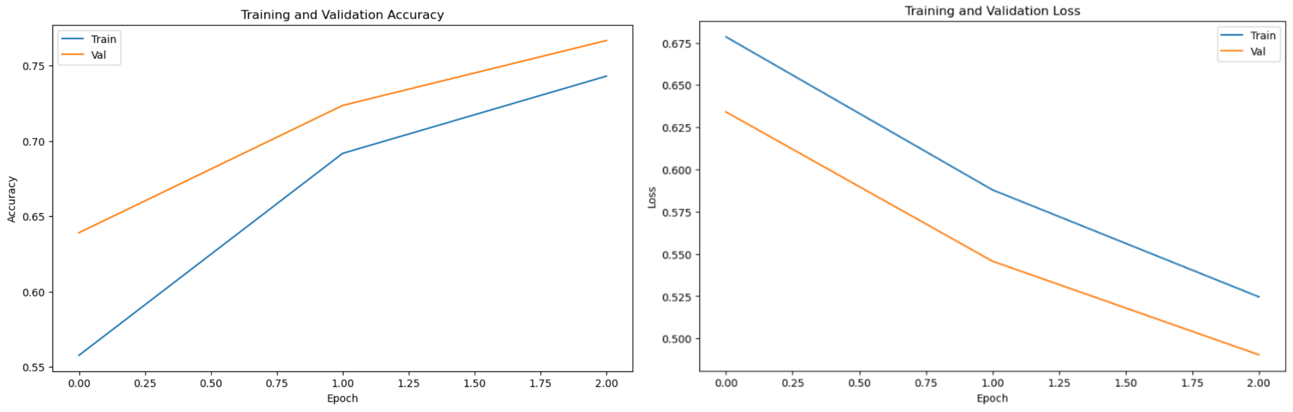


Figure 8. Model 2 with Dropout layer accuracy and loss plot

Model 2	**Train Accuracy**	**Train Loss**	**Validation Accuracy**	**Validation Loss**
Fold 1	0.6592	0.6016	0.7003	0.5673
Fold 2	0.6705	0.5895	0.7336	0.5300
Fold 3	0.6605	0.6021	0.7059	0.5637
Fold 4	0.6630	0.5958	0.7018	0.5662
Fold 5	0.6674	0.5962	0.7070	0.5565
Avg	**0.6641**	**0.5970**	**0.7097**	**0.5568**

Table 4. Model 1 with Dropout layer Training and validation results in each fold

According to the figure 5 and table 1 we can see that model 1 demonstrates consistent performance across the folds, achieving an average train accuracy of approximately 68.7% and an average validation accuracy of around 69.9% to 72.8%. The model effectively minimizes the discrepancy between predicted and actual labels, as indicated by an average train loss of 0.5593 to 0.5849 and an average validation loss of 0.5426 to 0.5751. It is worth noting that the inclusion of a dropout layer in Model 1 did not result in significant improvements, according to the figure 6 and table 2.

On the other hand, it can be seen in figure 7 and table 3 that model 2 achieves an average train accuracy of approximately 67.8% to 70.8% and an average validation accuracy of around 70.5% to 73.9%. The model effectively minimizes the discrepancy between predicted and actual labels, as indicated by an average train loss of 0.5486 to 0.5922 and an average validation loss of 0.5144 to 0.5638. Similar to model 1, the inclusion of a dropout layer in Model 2 does not noticeably enhance the results (figure 8 and table 4).

In summary, both models exhibit comparable performance, with moderate train and validation accuracies. Model 2 shows relatively better performance than model 1.

Conclusion

This project focused on developing Convolutional Neural Network (CNN) models for binary image classification of cats and dogs. The goal was to accurately classify images using different network architectures and training parameters. The dataset comprised 25,000 images, split evenly between dogs and cats. Various CNN models, including Model 1 and Model 2, were implemented and evaluated with different configurations of convolutional layers, pooling layers, and dense layers. A dropout layer was also introduced to assess its impact on model performance. The models were trained and validated using 5-fold cross-validation, and performance metrics such as accuracy and loss were measured.

The results demonstrated that the developed CNN models achieved high accuracy in classifying images of cats and dogs. Model 1 consistently performed well, with an average train accuracy of approximately 68.7% and an average validation accuracy ranging from 69.9% to 72.8%. The addition of a dropout layer did not significantly improve its performance. Model 2 also showed promising results, with an average train accuracy of 68.8% and an average validation accuracy ranging from 72.9% to 73.5%.

To further enhance the performance of the developed CNN model, future works can explore several areas. Firstly, expanding the dataset by collecting more diverse images of cats and dogs, including different breeds, poses, and backgrounds, would help the model generalize better to unseen data and improve its overall accuracy. Additionally, optimizing the CNN architecture by experimenting with different configurations, such as increasing network depth or width, and incorporating additional convolutional or pooling layers could lead to improved results. Fine-tuning hyperparameters and conducting an extensive search for the optimal combination can also contribute to enhanced performance. These future directions aim to refine the image classification model for cats and dogs and advance the field of CNN-based image classification.

References

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

Zhang, L., Lin, L., Liang, X., He, K., & Sun, J. (2016). Is faster R-CNN doing well for pedestrian detection. In European conference on computer vision (pp. 443-457). Springer, Cham.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

Jajodia, T., & Garg, P. (2019). Image classification–cat and dog images. *Image*, *6*(23), 570-572.

Lee, Y. (2021, January). Image classification with artificial intelligence: cats vs dogs. In *2021 2nd International Conference on Computing and Data Science (CDS)* (pp. 437-441). IEEE.

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.